Flexible QoS-Aware Service Composition in Highly Heterogeneous and Dynamic Service-Based Systems

Dionysios Efstathiou
Department of Informatics,
King's College London
dionysios.efstathiou@kcl.ac.uk

Peter McBurney
Department of Informatics,
King's College London
peter.mcburney@kcl.ac.uk

Steffen Zschaler
Department of Informatics,
King's College London
szschaler@acm.org

Johann Bourcier
IRISA, University of Rennes 1
johann.bourcier@irisa.fr

***Abstract*—Service-oriented pervasive systems, composed of a large number of devices with heterogeneous capabilities where devices' resources are abstracted as software services, challenge the creation of high-quality composite applications. Resource heterogeneity, dynamic network connectivity, and a large number of highly distributed service providers complicate the process of creating applications with specific QoS requirements. Existing approaches to service composition control the QoS of an application solely by changing the set of participating concrete services which is not suitable for ad-hoc service-based systems characterised by high intermittent connectivity and resource heterogeneity. In this paper, we propose a flexible way of formulating composition configurations suitable for such service-based systems. Our formulation proposes the combined consideration of the following factors that affect the QoS of a composed service: (a) service selection, (b) orchestration partitioning, and (c) orchestrator node selection. We show that the proposed formulation enables the definition of service composition configurations with 49% lower response time, 28% lower network latency, 36% lower energy consumption, and 13% higher success ratio compared to those defined with the traditional approach.**

I. Introduction

Service-oriented computing enables the creation of complex applications by composing services to provide functionality that none of the component services could provide by itself [15]. When configuring a service composition, apart from achieving functional goals, we also aim at creating composite applications which exhibit specified non-functional trade-offs.

Service-orientation is appropriate for engineering pervasive distributed systems composed of a large number of nodes with heterogeneous capabilities communicating over highly dynamic networks [2]. Nodes' resources such as application data, network connectivity, and hardware components are abstracted as loosely coupled software services [20]. An example of such a system is a fire-fighter decision support application where the goal is to combine services provided by heterogeneous devices such as sensors, and tablets, to compose complex applications for assisting fire-fighters to make well-informed decisions within a crisis.

Although several techniques have been proposed for service composition in traditional environments, most of them generally assume the existence of a single entity for orchestrating the interaction between services in a composition. *Service orchestrations* require strong assumptions of centralised control and fixed connectivity which are unrealistic when considering resource-constrained dynamic systems. To address these problems, more recent approaches proposed decentralised service orchestration [19]. However, these approaches do not consider system heterogeneity and dynamics when distributing the orchestration of composite applications or for providing applications that exhibit specified QoS.

We address these limitations by proposing a flexible approach for composing applications in distributed heterogeneous environments. Our approach considers three *degrees of freedom* (DoFs) for modifying a composition configuration: (a) service selection, (b) orchestration partitioning, and (c) orchestrator node selection. These DoFs enable us to formulate the space of all the possible combinations of different choices for realising a specific application.The proposed approach chooses flexibly the appropriate level of decentralisation for a composite application based on the resource availability of the participating nodes. Our simulation results show that our approach enables the formulation of composition configurations of higher quality than the traditional centralised orchestration. More specifically, we show that our approach outperforms the traditional way of composing services by enabling the definition of service composition configurations with 49% lower response time, 28% lower network latency, 36% lower energy consumption, and 13% higher success ratio.

The goal of this paper is to argue the need of multiple degrees of freedom when composing services in heterogeneous and dynamic environments and it is part of an overall framework which is described in [7].

The paper is organised as follows: Sect. II presents the background and discusses related work. Sect. III describes the motivating scenario of our research. Sect. IV describes our approach. Sect. V presents the experimental results of our study. Finally, Sect. VI presents conclusions and future work.

II. Background and Related Work

Service composition promotes the creation of complex applications by aggregating services to provide composite functionalities that none of the services could provide by itself. We use the concepts of *Concrete Service* which refers to an invocable service, and *Abstract Service* which defines abstractly the functionality of a service. An application can be seen as a composition of *Abstract Services* and can be represented as a *workflow* plan, as shown in Fig. 1. This plan defines how services interact with each other by specifying the order of services' invocation (control flow), and rules for data transfer between them (data flow).

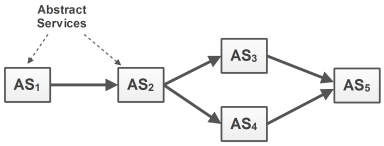


Fig. 1: An example abstract workflow plan

Currently, composition is synonymous with *Service Orchestration* where a central entity coordinates the control and data flow between participating services. On the other hand, *Service Choreography* defines the interaction protocol between several services from a global view with an emphasis on peer-to-peer collaboration.

Decentralised Orchestration [19] lies between these two extremes. With this approach, the coordination of the application is distributed to many nodes. Each orchestrating node integrates a local workflow engine and has only a partial view of the overall composition. These orchestrating nodes cooperate with each other towards realising the complete application.

A. QoS Models

In the current literature of service composition, there can be found two main groups of QoS models. The first family of QoS models concerns (web) services based on wired networks of fixed topology, and resource-rich service providers. These models present some generic QoS attributes [3], [6], [16], [21] for quantifying the quality of a provided single service and a service composition.

However, pervasive environments are characterised by resource heterogeneity, dynamic conditions, and user mobility. Several parameters, such as availability and performance, are difficult to define and evaluate in an end-to-end manner without considering the underlying environment (e.g. network, devices) which enables providers and users to interact and cannot be controlled by the provider [12]. For example, service availability depends on (i) provider availability, (ii) network availability, and (iii) user device availability. Thus, it is impossible to separate the QoS model from the environment dynamics when considering the end-to-end quality of a composition.

In this study, we focus on the second family of QoS models. Moorsel [13] tried to distinguish the quality of the provided service from the perceived quality from the user's perspective. In this study, the author makes a distinction between Quality of Experience (QoE), which are metrics as experienced by a service user, and QoS parameters of the system that provides a service and the provided service itself which are fully controllable by the service provider. QoE metrics may contain a subjective element in contrast to QoS measures, and QoE metrics may be influenced by any system between the service provider and the service user.

B. Service Composition Models

QoS-Aware Service Composition. Currently, optimising service composition is synonymous with the problem of *QoS-Aware Service Composition* [3], [4], [5], [6], [11]. In this problem, the goal is to find the set of concrete services to participate in a centralised orchestration that offer the required functionality, respect user's preferences and constraints, and optimise composition's QoS. The main limitation of this family of approaches is that they neglect to study the problems of centralised orchestration and try to optimise the quality of a composition by considering only the QoS of each service in isolation without considering how these services are composed together. For example, even if we select two services with very good QoS may not be enough indication about the quality of their composition if we neglect to consider the underlying network between them. Also, limited by the chosen QoS model, they neglect to take into account the underlying infrastructure when computing the QoS of a composition.

Distributed Service Composition. Decentralisation techniques aim at preserving the composition's operational semantics over different distributed configurations. SELF-SERV [19] is an extreme approach where the control of a composition is fully-decentralised among the participating services. Nanda and Karnik [14] proposed the idea of partitioning a centralised composition in decentralised fragments to improve system's scalability and concurrency. Schuhmann *et al.* [18] proposed a hybrid configuration of distributed applications in the context of pervasive environments by adjusting the suitable level of decentralisation based on the number of available resource-rich devices. Fdhila et al. [8] proposed a decentralised approach for composing applications into partitions of services that communicate frequently and by taking into account collocation and separation constraints towards optimising the overall QoS of the composition. However, the above approaches does not consider the degree of freedom of flexible placement of orchestrators for optimising the end-to-end QoS of a composition. Also, they fail to propose a method for achieving composite applications of specified quality which are optimal based on environmental conditions.

III. Use-case Scenario

Our scenario considers a fire-fighter decision support system that aims at improving the decision making of fire-fighters in an emergency situation.

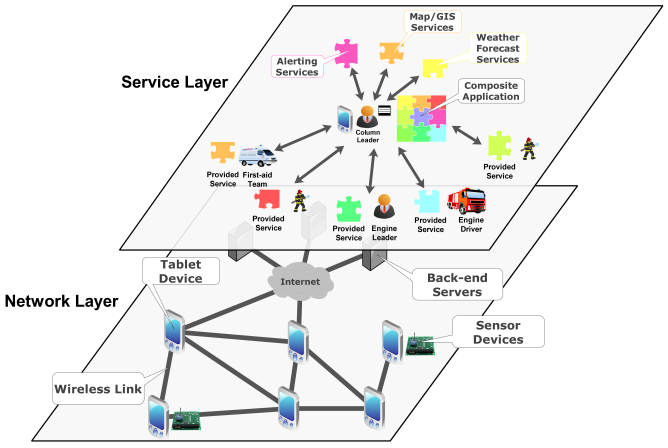


Fig. 2: The fire-fighting application scenario

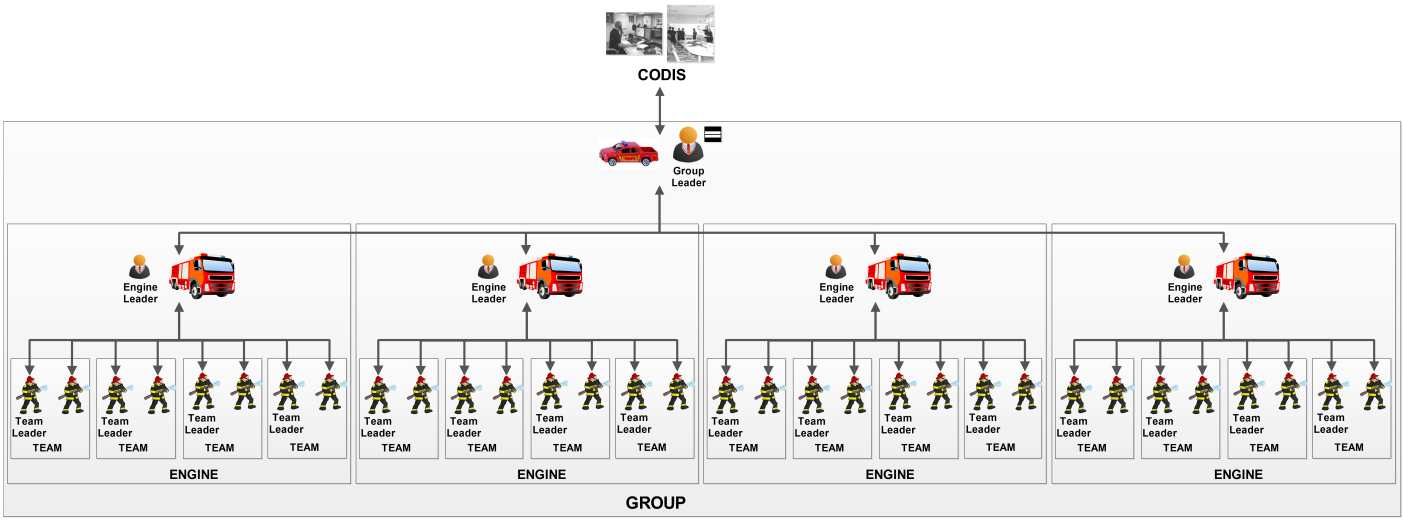


Fig. 3: Firefighter hierarchy in a forest fire scenario

Hierarchical Organization. The CODIS operational centre remotely supervises the actions of firefighters (FFs) and their resources, and acts as an advisor to the field commanding officer during emergency situations. According to firefighting operational rules, FFs are organised in groups of two to four people plus the team leader. Each firefighting force is dispatched into a sector and has responsibility for the assigned area. For example, Fig. 3 depicts a scenario with three hierarchical levels. In this example, FFs at the lowest level of hierarchy are called Team Members (TM) and they form pairs of TMs leaded by a Team Leader (TL). At a level above, an Engine Leader (EL) coordinates four TLs which comprise the crew of an engine. At the highest hierarchical level of our scenario, each Group Leader (GL) coordinates four ELs which form a Group of four engines.

Network and Services. Each FF carries a mobile device (e.g. tablet, mobile phone) and a number of special sensors (e.g. body and environmental temperature, accelerometer). Figure 2 presents the system architecture of our motivating scenario which is divided into two layers: network and service.

After the deployment of the FFs into the area of interest (e.g. forest) these devices form an infra-structureless Mobile Ad-Hoc Network (MANET). The maximum direct communication distance between two nodes is defined by their transmission range. Each device can communicate with neighbouring devices within their range, while the underlying routing protocol enables pair of distant devices to use services of each other even if there is no direct communication link between them by maintaining routing/communication paths.

Based on service-orientation principles, nodes' resources such as application data, network and hardware components, can be abstracted as loosely coupled software services [20]. The service layer refers to the description of services that represent resources provided by the various participating devices (e.g. tablets, sensor nodes, back-end servers). At this layer, composite applications are described as abstract workflow plans. The network layer is responsible for enabling devices to use services offered by other devices.

Service composition enables the realisation of complex applications by composing services provided by various devices. For example, consider the emergency situation of a forest fire. A commanding officer receives information from the fire-fighters about the position of the fire and local weather conditions. This data combined with information about the local geography of the area can be fed into prediction services for estimating the evolution of the fire. The commanding officer is able to take well-informed decisions based on this estimation and current availability of fire-fighting resources.

Our approach is not restricted to the presented scenario and can be applied to others exhibiting similar characteristics. Such applications can be found in the domain of *Mobile Enterprise Vision* [1], [17] where mobile devices offer and use remote services provided in a highly dynamic environment.

A. *Mobility Model of Firefighters*

We now provide the mobility models describing how the firefighters (FFs) move. The mobility model description is comprised of two main parts: (a) deployment, and (b) movement. The former defines how a FF of a specific hierarchical level is placed in the intervention area in relation to the position of the fire while the latter describes how the FF updates its position in relation to both the position of the fire and the group where the FF belongs to.

Deployment and Motion. FFs are deployed in hierarchical groups, where each group has a different purpose and mission to fulfil. We designed the mobility models of the various FF groups similarly to the main idea of the Reference Point Group Mobility (RPGM) [10] model which is widely used for simulating military battlefield communication scenarios. In RPGM, the nodes are divided into groups where each group has a logical center (leader) that determines the group's motion behaviour. Due to operational rules, FFs are organised into hierarchical groups where the motion of the members of a group is controlled or defined by the position of the leader. The parameters of the mobility models per FF hierarchical level chosen for our scenario are presented in Table I.

Mobile entities are continuously alternating between the $moving$ and $waiting$ states. Each time an entity decides to move it chooses a distance, direction, and speed of the movement. When the entity arrives at the new point, it waits until it moves again for a period of time calculated based on the minimum wait time and the variance of the wait time. The minimum wait time and the variance of the wait time per FF hierarchical level (Group, Engine, and Team) are denoted as $\_W\_M$ and $\_W\_V$ respectively in Table I.

Fire. The centre of the fire plays the role of the reference point for FFs' mobility models. Following the polar coordinate system, the position of a FF is defined by the centre of the fire (x_0, y_0), the distance d from the centre and the angle θ, as represented in Fig. 4. Thus, to define the position of a FF, we need to compute the parameters d, θ. In detail, the distance d of a FF is computed based on the following formula:

$$d\left(D_M, D_V\right) = R_{fire} + D_M + U\left[0, D_V\right] \tag{1}$$

where R_{fire} is the radius of the fire, D_M is the minimum distance of the FF from the fire, D_V is the maximum distance variation, and the uniform distribution $U\left[0, D_V\right]$ represents a randomly chosen distance variation of the FF from the fire.

While, the angle θ of a FF is computed as follows:

$$\theta\left(OFFSET, A_V\right) = OFFSET + U\left[0, A_V\right] \tag{2}$$

where A_V is the angle variation of the FF, the uniform distribution $U\left[0, A_V\right]$ represents the angle variation of the FF, and $OFFSET$ is determined by the angle of the sector of area which is assigned to the FF. Assume that the area around the centre of the fire is divided into four sectors and each sector has an angle of $\frac{\pi}{2}$. The first FF (e.g. group leader) is assigned to the first sector which spans from $\left[0, \frac{\pi}{2}\right]$ $(OFFSET = 0)$. The second FF is assigned to the second sector which spans from $\left[\frac{\pi}{2}, \pi\right]$ $(OFFSET = \frac{\pi}{2})$, and so on.

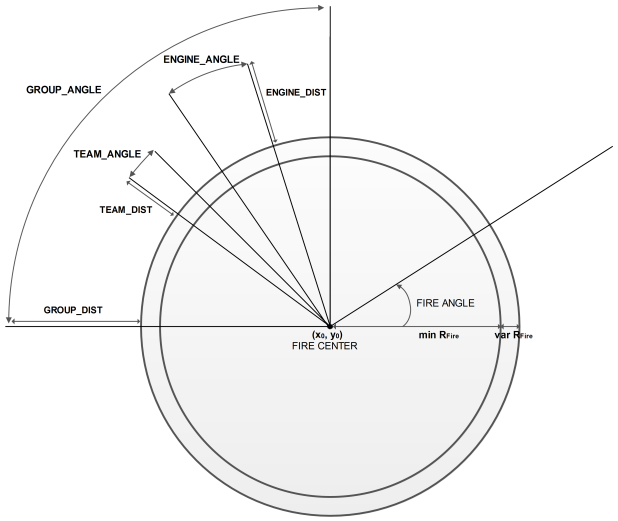


Fig. 4: Calculating the position of a firefighter

As mentioned previously, given the centre of the fire, we can define the position of a FF by computing the distance d from the centre and the angle θ as described in (1) and (2) respectively. We now describe the mobility model followed by the FFs in each hierarchical level by passing the appropriate parameters in the described formulas.

Parameter	Value	Parameter	Value	Parameter	Value
Group_D_M	50	Engine_D_M	30	Team_D_M	10
Group_D_V	20	Engine_D_V	10	Team_D_V	5
Group_A_V	$\pi/2$	Engine_A_V	$\pi/8$	Team_A_V	$\pi/16$
Parameter	Value	Parameter	Value	Parameter	Value
Group_S_M	15	Engine_S_M	5	Team_S_M	10
Group_S_V	10	Engine_S_V	0	Team_S_V	5
Group_W_M	120	Engine_W_M	60	Team_W_M	30
Group_W_V	60	Engine_W_V	30	Team_W_V	15

TABLE I: Parameters for deployment and motion of the simulated entities used in our experiments.

Group Leader. A Group Leader (GL) chooses a safe strategic position (large distance from fire) to be able to supervise the operation and have direct interaction with the his subordinates. The light all-road vehicle enables the GL to move quickly around the supervised ELs within the area of his supervision. These entities are moving with high minimum speed ($Group\_S\_M$) with medium direction changes ($Group\_S\_V$). These entities are not moving very frequently (high wait time - $Group\_W\_V$), but when they decide to move they perform large dislocations. More precisely, the distance d is computed based on (1) with parameters $d\left(Group\_D\_M, Group\_D\_V\right)$, and the angle θ based on (2) with parameters $d\left(OFFSET, Group\_A\_V\right)$.

Engine Leader. An Engine Leader (EL) is positioned closer to the fire than his supervising GL. The goal of the EL is to coordinate the team leaders under his supervision. These entities are moving with low minimum speed ($Engine\_S\_M$)with medium direction changes ($Engine\_S\_V$) which result to medium dislocations and they move more frequently than GLs ($Engine\_W\_V$). In detail, the distance d is computed based on (1) with parameters $d\left(Engine\_D\_M, Engine\_D\_V\right)$, and the angle θ based on (2) with parameters $d\left(OFFSET, Engine\_A\_V\right)$.

Team Leader/Member. The Team Leaders/Members (TLs/TM) are the actual FFs who are positioned very close to the fire. Most of the time, TL/TM are moving with medium minimum speed ($Team\_S\_M$) with high direction changes ($Team\_S\_V$) which result to small dislocations. TLs/TMs move the most frequently from all the other simulated entities ($Team\_W\_V$). In detail, the distance d is computed based on (1) with parameters $d\left(Team\_D\_M, Engine\_D\_V\right)$, and the angle θ based on (2) with parameters $d\left(OFFSET, Team\_A\_V\right)$.

B. QoS Metrics

We consider the following attributes for assessing the quality of a composite application configuration:

- *The response time* Q_{RT} of a configuration, which is the time from when a user issues a request until the user receives the result. This parameter is mainly affected by three factors: (a) the network round trip time (RTT) of the exchanged messages on the underlying network (Q_{RTT}); (b) the request processing time (RTS) that a service provider needs to process a request; and (c) the orchestration time (OT) for coordinating the execution of a set of services. To simulate

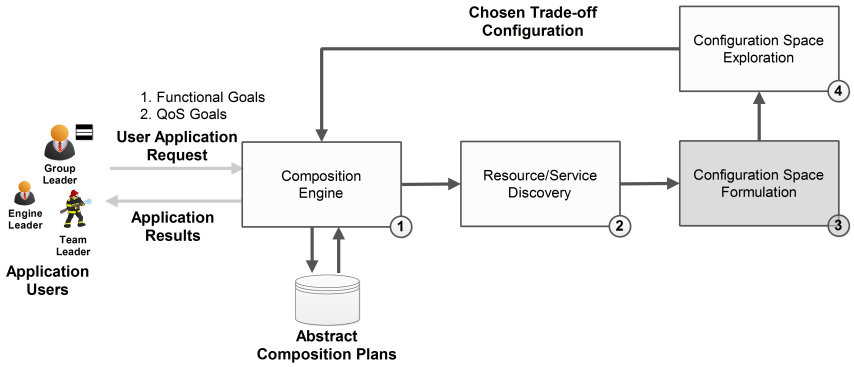


Fig. 5: Overview of our approach

the RTS of the services offered by the mobile devices in our scenario, we use an exponential distribution, as suggested in [9], with mean value equals to 1 (second) . We simulate the OT of an orchestrator node based on the following intuition: the more services are orchestrated by a node, the higher the orchestration time is. To achieve this, we multiply the value given by an uniform distribution with mean value equals to 1 (second) with the factor 2^n, where n is the number of orchestrators of a configuration.

- *The battery consumption* Q_{BC} of a configuration, which is the energy difference observed in the nodes participating in a configuration for realising a service composition configuration where nodes spent energy for: (a) sending/receiving data, and (b) orchestrating other services. We simulate the energy overhead of an orchestrator based on the following assumption: the more services are orchestrated by a node, the higher the amount of necessary energy is.
- *The success rate* Q_{SR} of a configuration, which is the fraction of successfully exchanged data between collaborating nodes within a composition configuration. Note that we did not use a reliable communication protocol to avoid the overhead of retransmissions which may be unrealistic for the studied scenario.

C. Problem Statement

In systems as the one above, assuming the existence of a central orchestrator is unrealistic due to the resource-constrained nature of the nodes, and their intermittent connectivity. Centralised orchestration leads to inefficient usage of resources because all intermediate data to be exchanged within a composition must be sent to the orchestrator. On the other hand, choreography distributes equally the control of an application among the participating nodes but it fails to exploit their resource heterogeneity. For example, some nodes may have higher resources available than others at a given time.

Two main groups of quality metrics can be identified when composing applications in such environments: (a) user-related goals (e.g. application response time); and (b) system-related goals (e.g. energy consumption). When considering multiple conflicting criteria, there is no single optimal configuration. Instead, the presence of multiple goals causes the creation of a set of trade-off configurations, known as Pareto-optimal. Without having further information about user's quality goals, none of these trade-off configurations can be said to be better than another. We consider a composition to be of *high-quality* when it maximises the satisfaction of user's QoS goals by simultaneously having minimal impact on the underlying system. The goal of our research is to provide a flexible formulation of composition configurations that are functionally equivalent and exhibit good non-functional trade-offs between the user-related and system-related conflicting goals.

IV. The Proposed Approach

We now describe the overview of the proposed approach, the considered degrees of freedom, an example composition configuration and a meta-model for defining composition configurations in highly distributed service-based environments.

A. High-level View

Figure 5 highlights with dark colour the focus of this paper and presents the overview of the process of composing applications in highly distributed heterogeneous environments. Firstly, the users submit a request along with a set of quality goals for a composite application whose abstract plan is already stored in the composition engine. Then, the service/resource discovery routine discovers the resources which are available for realising the requested application. Then, the space of all possible realisation configurations is formulated. Finally, an optimisation process searches this design space of possible configurations and returns the configuration which satisfies the best the requested goals.

B. Degrees of Freedom

We call a *degree of freedom* (DoF) a parameter of a composition configuration which is free to be varied to affect the QoS of the realised application while leaving its functional part unaffected. After designing the abstract plan of a composite application, there are many DoFs of various types which form a large design space of configurations of shared functionality but each of which differs in their QoS trade-offs. In other words, DoFs enable us to formulate the design space which includes all the possible combinations of different choices for realising a specific application. We call the set of choices for realising a composite application, *composition configuration*.

The set of all possible configurations is called the *design space*, which has as many dimensions as the possible design options.

We group the possible types of changes in a composition configuration into three DoFs: (a) which concrete services are selected for implementing the $Abstract\ Services$ of the composition; (b) what is the partitioning of the workflow to sub-orchestrations; and (c) in which $Nodes$ to deploy the selected sub-orchestrations. Current approaches consider only the first DoF. Below we describe in detail the considered DoFs.

Service Selection. For each abstract service which describes a necessary functionality, there may exist many concrete services that satisfy the same functionality possibly with different QoS properties. Based on the desired QoS goals (e.g. the response time must be lower than an upper threshold) together with information about the offered QoS of the services, these concrete services can be used interchangeably in a composition. Fig. 6 shows three concrete services provided by different nodes that implement the same abstract service, where the goal is to choose one to participate in the composition.

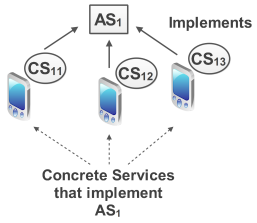


Fig. 6: The first DoF: service selection

Orchestration Partitioning. In dynamic resource-constrained environments, it is unrealistic to assume that a resource-rich central orchestrator will be able to reach all the services to participate in an application. On the other hand, completely distributed orchestration does not provide any point of control for configuring applications that exhibit specific quality goals. Thus, we propose to flexibly decentralise the application's orchestration into sub-orchestrations that will exploit the benefits from both extreme approaches. In detail, the initial plan is partitioned into sub-workflows that respect the control flow of the parent plan. Thus, this approach is able to produce plans with various levels of decentralisation: from fully centralised orchestration to fully decentralised. We expect that hybrid models may optimise the composition performance based on run-time conditions. Fig. 7 shows an example partitioning of the initial plan into two sub-workflows.

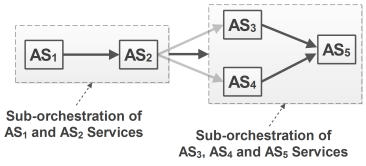


Fig. 7: The second DoF: workflow partitioning

Orchestrator Node Selection. After having partitioned the initial plan into sub-workflows, the last step is to assign the sub-workflows to orchestrating nodes to realise them. Fig. 8 depicts an example assignment of the two sub-workflows based on the previous example to orchestrating nodes that will be responsible for realising the relevant sub-orchestrations. The choice of an orchestrating node can depend on its available resources such as battery level, node availability, etc. For instance, nodes with limited resource availability, or nodes that do not have direct access to all the services to be orchestrated, are not good orchestration candidates.

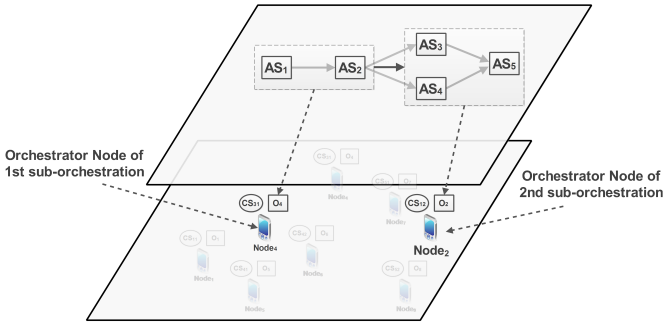


Fig. 8: The third DoF: sub-orchestrator selection

An example composition configuration. To realise an application such as the one in Fig. 1, we have to make a choice for each described DoF. In Fig. 9 we depict an example configuration where the following concrete services were chosen to participate in the composition: CS_{12}, CS_{21}, CS_{31}, CS_{42}, and CS_{51}. Also, we decided to group the abstract services into the two following sub-orchestrations: $suborch_1 = \langle AS_1, AS_2 \rangle$ and $suborch_2 = \langle AS_3, AS_4, AS_5 \rangle$. Finally, we chose $Node_2$ and $Node_4$ to coordinate these sub-orchestrations.

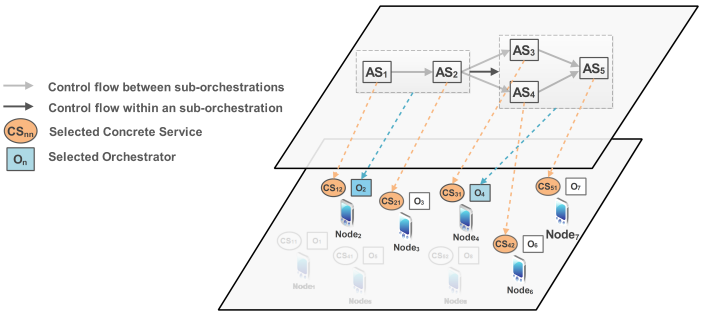


Fig. 9: An example composition configuration

V. Evaluation

In this section we seek to answer the following research question: "Does the proposed model make available configurations of higher QoS than the traditional approach?"

A. Simulated Scenario

We simulate a service-based firefighter decision support system where firefighters of three different hierarchical levels (Group, Engine, and Team) carry devices which offer software services and cooperatively form an infrastructureless MANET. According to the operational rules described in Section III, we consider the following hierarchical formation: (a) 2 Group Leaders, (b) 8 Engine Leaders, and (c) 32 Team Leaders/Members, which results in a network of 42 nodes. To

simulate the MANET formed by firefighters' devices, we used the well-known Network Simulator 3 (NS-3) [1].

Table II shows the settings chosen for the network-related aspects of our simulation. Firstly, we configured 42 mobile devices with wireless capabilities which form an infrastructure-less MANET where each device can communicate with other devices within their proximity, while the underlying routing protocol enables pairs of distant devices to communicate with each other even if there is no direct link between them. Node mobility causes unpredictable making/breaking of links as nodes can move out of range from one another at any-time and vice versa. To realistically capture the effects of firefighters mobility which results in unexpected topology changes on the performance of the simulated service-based system, we integrated into NS-3 the mobility models described in Section III-A. Finally, we used the models provided by NS-3 for simulating the energy consumption at a particular node.

During the emergency situation, the Group Leaders request the results of the complex application shown in Fig. 1. Firefighters' devices offer concrete services which implement the abstract services of the presented composition plan. The underlying network enables users to call services offered by other devices (providers). We simulate a composite application as a collection of service interactions coordinated by orchestrator nodes. An orchestrator is responsible for coordinating the execution of a composition by calling services, aggregating their intermediate results and sending the final results to the user who issued the request for the composite application.

Number of nodes	42
Wifi standard	80211b
Wifi rate	DsssRate1Mbps
Transmission range (R)	45 m
Routing protocol	DSDV
Protocol stack	UDP/IPv4
Propagation loss model	Log-Distance/$\alpha 3$
Number of concrete services	42
Size of composition plan	5 (Abstract Services)
Data packet size	1 KB
Request packet size	1 KB

TABLE II: Wireless network simulation parameters.

B. Results and Discussion

We compare the two composition approaches: (a) the traditional approach that considers only the service selection DoF; and (b) our proposed formulation which takes into account three DoFs. The size of the configuration space formulated by the two approaches is $\sim 2 \cdot 10^4$ and $\sim 6 \cdot 10^7$ configurations respectively. Simulating a single configuration takes ~ 500 secs in a machine with Intel Core i7 vPro with 12GB DDR3 RAM, running the Linux 3.2.0-40 kernel. This corresponds to ~ 11 and $\sim 2 \cdot 10^5$ years which makes the simulation of the entire space infeasible. To avoid this obstacle and answer our research question, we randomly sampled each search space and compared the quality of 1000 random configurations.

By using a fixed seed, we simulated all the configurations in the same environment to study how different configurations affect the considered QoS metrics. Moreover, we simulated each configuration 50 times, allowing the network to evolve in accordance with the described mobility models, with 30 seconds between each of the 50 runs. We do this to study the effect of network dynamics on performance of the composition configuration. Each run takes approximately 1800 simulation seconds in total. To ensure that the network is completely configured before simulating a composition configuration we included a set-up/warm-up time of 20 seconds.

Figures 10(a) - 10(d) show the results of our experiment per each of the four quality dimensions of interest: (a) response time, (b) network latency, (c) energy consumption, and (d) success rate. Fig. 10(a) shows that our approach achieves lower response time. This can be explained by the fact that in the traditional approach all intermediate application data must be routed through the centralised orchestrator to the participated nodes which results to higher response time (Q_{RT}) than our decentralised approach. The response time of a composition is affected by two main factors: (a) the orchestration overhead, and (b) the network response time between interacting services. As mentioned earlier, the overhead for coordinating an orchestration increases as the number of orchestrating services increases. In other words, the existence of a single node for orchestrating all concrete services leads to the highest possible overhead. For the second factor (b), the traditional approach results in higher network response time as shown in Fig. 10(b). The traditional case where each node communicates with a single central entity results in routing paths of higher length, and thus higher response time, in contrast to our approach which enables the flexible placement of orchestrators close to the called services. Note that Q_{RTT} is computed based on the network latency of only the successfully delivered messages.

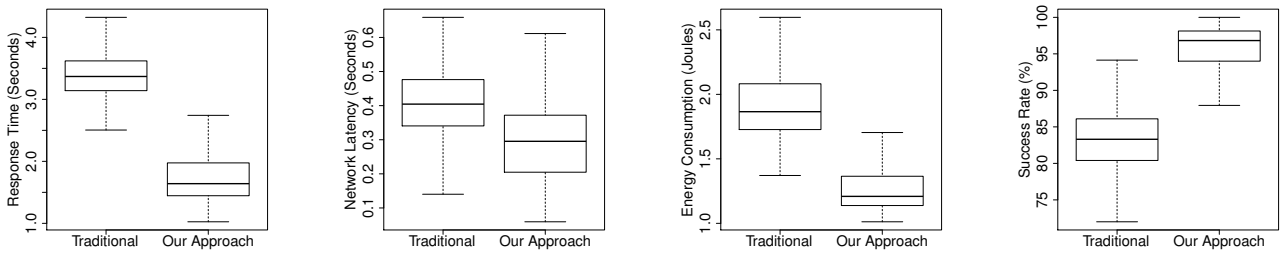
The previous fact affects also the Q_{SR} metric whose results are depicted in Fig. 10(d). In detail, due to the longer routing paths, the traditional approach achieves a lower success rate as the probability of losing a packet increases with the length of the communication path. Finally, concerning the energy consumption (Q_{BC}) in Fig. 10(c), the traditional approach consumes more energy due to: (a) the resource overhead of an orchestration increases as the number of coordinated services increases, and (b) the communication overhead is higher for receiving/transmitting application data due to the fact that all intermediate data pass through the centralised orchestrator.

Table III shows the numerical results for the considered QoS metrics for both the composition models in comparison. The table lists the mean, standard deviation, median, min, and max values of 1000 simulated configurations each one repeated 50 times. We use the non-parametric Mann-Whitney test to evaluate statistical significance because we have no information about the distribution of the data. The null hypothesis (H_0) states that the composition approaches produce configurations of the same quality. H_0 is rejected by the Mann-Whitney test at 1% significance level (p-value $< 2.210^{-16}$).

VI. Conclusions and Future Work

We have presented a flexible formulation of the service composition configuration space, which is suitable for the highly distributed and heterogeneous nature of mobile cloud systems. Our formulation differs significantly from prior work in the area of composition optimisation, which mostly relies on centralised orchestration and the limited degree of freedom of service selection. In contrast, our approach promotes the idea

[1] http://www.nsnam.org/



(a) Response Time Q_{RT} (b) Round Trip Time Q_{RTT} (c) Consumed Energy Q_{EC} (d) Success Rate Q_{SR}

Fig. 10: Varying the Size of Problem Dimensions

		Composition Method		
QoS Metric		Traditional	Proposed	p-value
Q_{RT} (S)	Mean	3.393	**1.748**	$<2.2\ 10^{-16}$
	SD	0.359	0.433	
	Min	2.345	1.025	
	1st Qu.	3.141	1.446	
	Median	3.369	**1.640**	
	3rd Qu.	3.620	1.975	
	Max	4.435	3.700	
Q_{RTT} (S)	Mean	0.40930	**0.293**	$<2.2\ 10^{-16}$
	SD	0.098	0.110	
	Min	0.099	0.059	
	1st Qu.	0.341	0.2048	
	Median	0.404	**0.295**	
	3rd Qu.	0.476	0.372	
	Max	0.754	0.651	
Q_{BC} (J)	Mean	2.032	**1.311**	$<2.2\ 10^{-16}$
	SD	0.509	0.281	
	Min	1.371	1.012	
	1st Qu.	1.727	1.138	
	Median	1.866	**1.209**	
	3rd Qu.	2.082	1.364	
	Max	4.274	3.068	
Q_{SR} (%)	Mean	82.69	**95.50**	$<2.2\ 10^{-16}$
	SD	4.954	4.193	
	Min	62.57	65.88	
	1st Qu.	80.40	94.00	
	Median	83.30	**96.82**	
	3rd Qu.	86.10	98.13	
	Max	95.25	100.00	

TABLE III: Comparing the quality of composite applications.

of flexibly decentralising the orchestration task of a composite application into sub-orchestrations and selecting the appropriate nodes for orchestrating the resulting sub-orchestrations.

We plan to compare the efficiency of different optimisation algorithms for searching the formulated design space. More research is needed to study the performance of these algorithms when various system changes may happen at run-time. Finally, we plan to evaluate the performance of our solution by using sophisticated simulation tools and real-world datasets.

Acknowledgements. This work has been supported by the European FP7 Marie Curie Initial Training Network "RELATE" (Grant Agreement No. 264840).

REFERENCES

[1] Enterprise Mobility Leveraging the Mobility Economy in the Enterprise. White paper, Hewlett-Packard Development Company, Februrary 2012.

[2] J. Bourcier and C. Escoffier. Implementing Home-Control Applications on Service Platform. In *4th CCNC*, 2007.

[3] G. Canfora, M. Di Penta, R. Esposito, and M. L. Villani. An Approach for QoS-Aware Service Composition Based on Genetic Algorithms. In *GECCO*, 2005.

[4] V. Cardellini, E. Casalicchio, V. Grassi, S. Iannucci, F. L. Presti, and R. Mirandola. MOSES: A Framework for QoS Driven Runtime Adaptation of Service-Oriented Systems. *TSE*, 99, 2011.

[5] D. Chakraborty, A. Joshi, T. Finin, and Y. Yesha. Service Composition for Mobile Environments. *MONET*, 10(4):435–451, Jan. 2005.

[6] D. B. Claro, P. Albers, and J. kao Hao. Selecting Web Services for Optimal Composition. In *Proc. of 2nd SDWP*, 2005.

[7] D. Efstathiou, P. McBurney, S. Zschaler, and J. Bourcier. Exploring Optimal Service Compositions in Highly Heterogeneous and Dynamic Service-Based Systems. In *5th SSBSE*, pages 313–318, 2013.

[8] W. Fdhila, M. Dumas, C. Godart, and L. Garca-Bauelos. Heuristics for Composite Web Service Decentralization. *SoSyM*, pages 1–21, 2012.

[9] A. Gorbenko, V. Kharchenko, S. Mamutov, O. Tarasyuk, Y. Chen, and A. Romanovsky. Real Distribution of Response Time Instability in Service-Oriented Architecture. In *29th IEEE Symposium on Reliable Distributed Systems*, pages 92–99, 2010.

[10] X. Hong, M. Gerla, G. Pei, and C. chuan Chiang. A Group Mobility Model for Ad Hoc Wireless Networks. pages 53–60, 1999.

[11] N. B. Mabrouk, S. Beauche, E. Kuznetsova, N. Georgantas, and V. Issarny. Qos-aware service composition in dynamic service oriented environments. In *Middleware*, 2009.

[12] C. Marchetti, B. Pernici, and P. Plebani. A Quality Model for Multichannel Adaptive Information Systems. In *13th International World Wide Web Conference*, pages 48–54, 2004.

[13] A. V. Moorsel. Metrics for the Internet Age: Quality of Experience and Quality of Business. Technical report, Performability Workshop, 2001.

[14] M. G. Nanda. Synchronization analysis for decentralizing composite web services. *IJCIS*, 13:91–119, 2004.

[15] M. Papazoglou and D. Georgakopoulos. Introduction: Service-Oriented Computing. *Comm. of ACM*, 46:24–28, 2003.

[16] F. Rosenberg, C. Platzer, and S. Dustdar. Bootstrapping Performance and Dependability Attributes of Web Services. In *IEEE International Conference on Web Services*, pages 205–212, 2006.

[17] SAP. Realize the Full Potential of Enterprise Mobility, 2011.

[18] S. Schuhmann, K. Herrmann, and K. Rothermel. Efficient Resource-Aware Hybrid Configuration of Distributed Pervasive Applications. In *the 8th PerCom*, 2010.

[19] Q. Z. Sheng, B. Benatallah, M. Dumas, and E. O. yan Mak. SELF-SERV: A Platform for Rapid Composition of Web Services in a Peer-to-Peer Environment, 2002.

[20] J. a. P. Sousa and D. Garlan. Aura: An Architectural Framework for User Mobility in Ubiquitous Computing Environments. In *Proc. of the 3rd WICSA*, 2002.

[21] L. Zeng, B. Benatallah, A. Ngu, M. Dumas, J. Kalagnanam, and H. Chang. QoS-Aware Middleware for Web Services Composition. *IEEE Transactions on Soft. Eng.*, pages 311 – 327, 2004.